Journal of Discrete Algorithms 7 (2009) 249–255

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Improved approximation algorithms for the single-sink buy-at-bulk network design problems ☆

Raja Jothi [a,*], Balaji Raghavachari [b]

[a] Biostatistics Branch, National Institute of Environmental Health Sciences, National Institutes of Health, Research Triangle Park, NC 27709, USA
[b] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA

ARTICLE INFO

Article history:
Received 4 November 2006
Accepted 19 December 2008
Available online 31 January 2009

Keywords:
Single-sink buy-at-bulk
Network design
Approximation algorithms
Randomized algorithms
Combinatorial optimization

ABSTRACT

Consider a given undirected graph $G = (V, E)$ with non-negative edge lengths, a root node $r \in V$, and a set $D \subseteq V$ of *demands* with d_v representing the units of flow that demand $v \in D$ wishes to send to the root. We are also given K types of cables, each with a specified capacity and cost per unit length. The *single-sink buy-at-bulk* (SSBB) problem asks for a low-cost installation of cables along the edges of G, such that the demands can simultaneously send their flow to root r. The problem is studied with and without the restriction that the flow from a node must follow a single path to the root. We are allowed to install zero or more copies of a cable type on each edge. The SSBB problem is NP-hard. In this paper, we present a 153.6-approximation algorithm for the SSBB problem improving the previous best ratio of 216. For the case in which the flow is splittable, we improve the previous best ratio of 76.8 to α_K, where α_K is less than 67.94 for all K. In particular, $\alpha_2 < 17.7$, $\alpha_3 < 23.2$, $\alpha_4 < 28.8$, and $\alpha_5 < 34.3$.

Published by Elsevier B.V.

1. Introduction

Consider a given undirected graph $G = (V, E)$ with non-negative edge lengths, a root node $r \in V$, and a set $D \subseteq V$ of *demands* with d_v representing the units of flow that demand $v \in D$ wishes to send to the root. We are also given K types of cables, each with a specified capacity and a cost per unit length. The cost per unit capacity per unit length of a high-capacity cable is typically less than that of a low-capacity cable, reflecting "economy of scale". In other words, it is cheaper to buy a cable of larger capacity than many cables (adding upto same capacity) of smaller capacity. The extensively studied *single-sink buy-at-bulk* (SSBB) problem, also known as the *single-sink edge installation* problem, asks for a low-cost installation of cables along the edges of G, such that the demands can simultaneously send their flows to root/sink r under the restriction that the flow from a node must follow a single path to the sink, i.e., the flow from a node to the root is *unsplittable*. We are allowed to install zero or more copies of a cable type on each edge. By *divisible* SSBB (DSSBB) problem, we refer to the version of the SSBB problem in which the flow is divisible/splittable.

The SSBB problem has applications in the hierarchical design of telecommunication networks, in which the traffic from a source must follow a single path to the sink. The DSSBB problem has its own applications: a classic application would be that of routing oil from several oil wells to a major refinery [11].

The *buy-at-bulk* network design problem was introduced by Salman et al. [11]. They showed that the problem is NP-hard through a simple reduction from the Steiner tree problem. The problem remains NP-hard even when only one cable

☆ Preliminary version of this paper appeared in Proc. SWAT 2004 [R. Jothi, B. Raghavachari, Improved approximation algorithms for the single-sink buy-at-bulk network design problems, in: Proc. 9th Scandinavian Workshop on Algorithm Theory (SWAT), 2004, pp. 336–348. [8]].

* Corresponding author.
E-mail address: jothi@mail.nih.gov (R. Jothi).

1570-8667/$ – see front matter Published by Elsevier B.V.
doi:10.1016/j.jda.2008.12.003

type is available. They also presented a $O(\log n)$-approximation algorithm for the SSBB problem in Euclidean graphs. For problem instances in general metric spaces, Awerbuch and Azar [1] presented a $O(\log^2 n)$-approximation algorithm. Their algorithm works even for multi-root version of the problem. Bartal's tree embeddings [2] can be used to improve their ratio to $O(\log n \log\log n)$. The ratio can be improved further to $O(\log n)$ using Fakcharoenphol et al.'s improved results for tree embeddings [3]. Garg et al. [5] presented an $O(K)$-approximation algorithm based on LP-rounding. Guha, Meyerson and Munagala [6] presented the first constant-factor approximation algorithm, whose ratio was estimated to be around 2000 by Talwar [12]. In the same paper, Talwar presented an LP-based rounding algorithm with an improved ratio of 216.

Recently, Gupta, Kumar and Roughgarden [7] presented a simple and elegant 76.8-approximation[1] algorithm for the SSBB problem. But unfortunately, their approach does not guarantee that the flow from a node follow a single path to the sink. In other words, their ratio of 76.8 holds for the DSSBB problem, but not for the SSBB problem. That leaves Talwar's ratio of 216 as the current best for the SSBB problem.

For the closely related *Capacitated Minimum Steiner Tree* (CMStT) problem with just one cable type, in which the final solution is required to be a minimum cost tree network, the current best approximation ratio is $\gamma\rho + 2$, due to Jothi and Raghavachari [9], where ρ is the best achievable approximation ratio for the minimum Steiner tree problem, and γ is the inverse Steiner ratio.

In this paper, we present a 153.6-approximation algorithm for the SSBB problem. We also propose a modification to Gupta et al.'s DSSBB algorithm that reduces the ratio from 76.8 to α_K, where α_K is less than 67.94 for all K. In particular, $\alpha_2 < 17.7$, $\alpha_3 < 23.2$, $\alpha_4 < 28.8$, and $\alpha_5 < 34.3$. At the time this paper was in review, Grandoni and Italiano [4] have shown that Gupta et al.'s DSSBB algorithm actually provides a 64.8 approximation. They have also shown a much better 24.92 approximation ratio for the DSSBB problem using a simpler algorithm based on [7].

2. Preliminaries

Let $G = (V, E)$ be the input graph with $D \subseteq V$ denoting the set of demands. We use the terms vertices and nodes interchangeably. Also, depending upon the context, we use the term "demand" to denote a vertex or the flow out of it. The term "weight" is exclusively used to denote the flow out of a vertex. Let c_e denote the length of edge e. We also use c_{xy} to denote the length of an edge connecting nodes x and y. We use the metric completion of the given graph. Let u_i and σ_i denote the capacity and cost per unit length of cable type i. We define $\delta_i = \sigma_i/u_i$ to be the "incremental cost" of using cable type i. The value of δ_i can also be interpreted as the cost per unit capacity per unit length of cable type i. Let us assume that each u_i and σ_i (and by definition δ_i) is a power of $1+\epsilon$, $\epsilon > 0$, which can be enforced by rounding each capacity u_i down to the nearest power of $1+\epsilon$, and each σ_i upto the nearest power of $1+\epsilon$. This assumption is not without loss of generality, and can be accounted by losing a factor of $(1+\epsilon)^2$ in the approximation ratio. We will choose ϵ later. This idea of rounding is adapted from [7], in which they used powers of 2, thus effectively choosing ϵ to be 1.

The following properties on the costs and capacities of cable types have been known [6,7]. Without loss of generality, assume that the cables are ordered such that $u_i < u_j$ and $\sigma_i < \sigma_j$ for all $i < j$. Note that if $u_i \leqslant u_j$ and $\sigma_i \geqslant \sigma_j$, then we can eliminate cable type i from consideration. We can also assume that $u_1 = \sigma_1 = 1$, as this can be obtained by appropriate scaling, though it may leave non-integer weights at vertices. Since the cable costs reflect economy of scale, for each $j < k$,

$$\frac{\sigma_k}{u_k} < \frac{\sigma_j}{u_j}. \tag{1}$$

The fact that $\delta_j = \sigma_j/u_j$ is a power of $1+\epsilon$ implies that $\delta_{j+1} \leqslant \delta_j/(1+\epsilon)$ for all j. Let $g_k = \frac{\sigma_{k+1}}{\sigma_k} u_k$. By Eq. (1),

$$1 = u_1 < g_1 < u_2 < g_2 < \cdots < u_K < g_K = \infty.$$

Since σ_i is a power of $1+\epsilon$ for any i, and $\sigma_{j+1} > \sigma_j$, using Eq. (1) we get,

$$\frac{u_{j+1}}{u_j} > 1+\epsilon.$$

Let OPT denote an optimal solution with cost $C^* = \sum_j C^*(j)$, where $C^*(j)$ is the amount paid for cable type j in OPT. We state the following lemma and its proof from [7], as its understanding is crucial for an easier understanding of our algorithms.

Lemma 2.1 *(Redistribution Lemma [7]). Let T be a tree rooted at r with each edge having capacity U. For each vertex $j \in T$, let $w(j) < U$ be the weight located at j with $\sum_j w(j)$ a multiple of U. Then there is an efficiently computable (random) flow on the tree that redistributes weights without violating edge capacities, so that each vertex receives a new weight $w'(j)$ that is either 0 or U. Moreover,*

$$\mathbf{Pr}\big[w'(j) = U\big] = w(j)/U.$$

[1] Although the ratio reported in [7] is 72.8, the ratio must actually be 76.8 (A. Gupta and T. Roughgarden, personal communication).

Proof. Replace each edge in T with two oppositely directed arcs. Let Y be a value chosen uniformly at random from $(0, U]$. Take an Euler tour of the vertices in T, starting from r and visiting all the other vertices $\{j_1, j_2, \ldots, j_m\}$ in T. Let a counter Q be set to 0 initially. On visiting vertex j_k, we update $Q \leftarrow Q + w(j_k)$. Also, let Q_{old} and Q_{new} be the value of Q just before and after visiting j_k, respectively. On visiting j_k, if $xU + Y \in (Q_{old}, Q_{new}]$ for some integer x, then "mark" j_k and ask that it sends $Q_{new} - (xU + Y)$ weight to the next marked vertex lying clockwise on the tour. In the other case, we ask that j_k sends all its weight to the next marked vertex lying clockwise on the tour. This construction ensures that the maximum flow on a directed edge is at most U, and that the probability that a vertex j gets marked is $w(j)/U$, which is exactly the probability that j receives a weight of U.

Since we are working on a directed tour, the cost of this redistribution is at most twice the cost of the tree T, as an edge in T was replaced by two oppositely directed arcs. But, using simple flow canceling argument, one can show that one copy of the edges in T is sufficient for such a redistribution. □

3. Algorithms

We first present our main result—an approximation algorithm for the SSBB problem that achieves an approximation ratio of 153.6. In Section 3.2, we propose a minor modification to Gupta et al.'s DSSBB algorithm that reduces the ratio from 76.8 to α_K, where α_K is less than 67.94 for all K. In particular, $\alpha_2 < 17.7$, $\alpha_3 < 23.2$, $\alpha_4 < 28.8$, and $\alpha_5 < 34.3$.

3.1. The SSBB problem

Let $G = (V, E)$ be the input graph with root $r \in V$, and let $D \subseteq V$ be the set of demands with d_j denoting the weight at j. Recall that the vertices in D may have non-integer weights because of the scaling done to ensure $u_1 = \sigma_1 = 1$. To guarantee integral demands at the vertices, Gupta et al. [7] first construct T_0, a ρ-approximate Steiner tree spanning D, using cables of capacity u_1. They then redistribute the demands using the construction in the proof of Lemma 2.1 with $U = u_1$ to collect *integral* demands at some subset of vertices in D. Later, vertices of integral demands are duplicated so that the demands at vertices are unit weight. Because of this redistribution and duplication, there is no guarantee that the demand from a vertex in the input graph travels along a single path to the sink, as the demand at a vertex may have been split during the redistribution and/or duplication process. In other words, this redistribution strategy (Lemma 2.1) does not guarantee the indivisibility of flow required for SSBB problem.

In our algorithm below, we make sure that the demand at a vertex follows a single path to the sink. Like in [7], we set $\epsilon = 1$, which makes u_i and σ_i (and by definition δ_i) powers of 2. This gives us the flexibility of generating u_{i+1} weighted nodes from integral number of u_i weighted nodes, thereby eliminating splitting of demands. First, we introduce a new "redistribution" procedure, which is pivotal in guaranteeing that the flow from a source follows a single path to the sink. Recall that Lemma 2.1 redistributes the weights uniformly at random, and the probability that a vertex receives a weight of U is proportional to its weight.

Lemma 3.1. *Either there exists at least one arc with zero flow in the directed tour t constructed in the procedure of Lemma 2.1, or there exists a redistribution (using Lemma 2.1), with zero flow on at least one arc of the directed tour, which produces the exact same assignment of weights.*

Proof. The proof is complete if the first part of the lemma were true. Suppose it were not true. Let t be the directed tour in the procedure of Lemma 2.1, which was used to redistribute the weights. Let $m > 0$ be the smallest flow across a directed edge in t. Note that $m \leqslant U$. For each directed edge in t, subtract m from the flow on that edge. After this, we are guaranteed that at least one edge in t has a zero flow. The fact that this post-processing does not alter the distribution of weights completes the proof. □

Lemma 3.2. *There exists a redistribution using the procedure of Lemma 2.1 with $Y = U$, which produces the exact same assignment of weights as that with Y that is chosen uniformly at random from $(0, U]$.*

Proof. Let t be the directed tour in the proof of Lemma 2.1. As per Lemma 3.1, there exists at least one edge in t with zero flow. Let e be an edge, from vertex p to vertex q, in t with zero flow. Without loss of generality, we can assume that $p \in D$. As per the construction in the proof of Lemma 2.1, p must be one of the vertices that must have been marked. Since the flow on e is zero, it must be that Q_{new} at p is equal to $xU + Y$ for some integer x, which means that vertex g marked just after p must either have $(x+1)U + Y \in (Q_{old}$ at g, Q_{new} at $g]$ or $Y \in (Q_{old}$ at g, Q_{new} at $g]$. This means that Q_{new} at g is at least U greater than Q_{new} at p.

Recall from the proof of Lemma 2.1 that the vertices in t are visited starting from r. We now show that a construction with $Y = U$ on t, visiting vertices starting from q (instead of r) produces the exact same assignment of weights as that with Y that is chosen uniformly at random from $(0, U]$. From the above discussion, since Q_{new} at g is at least U greater than Q_{new} at p, and the flow on e is zero, it can be seen that the construction with $Y = U$ on t and visiting vertices starting from q produces the exact same outcome as what is desired, i.e., the set of vertices that were assigned a weight of U will exactly be the same as that marked in the proof of Lemma 2.1. □

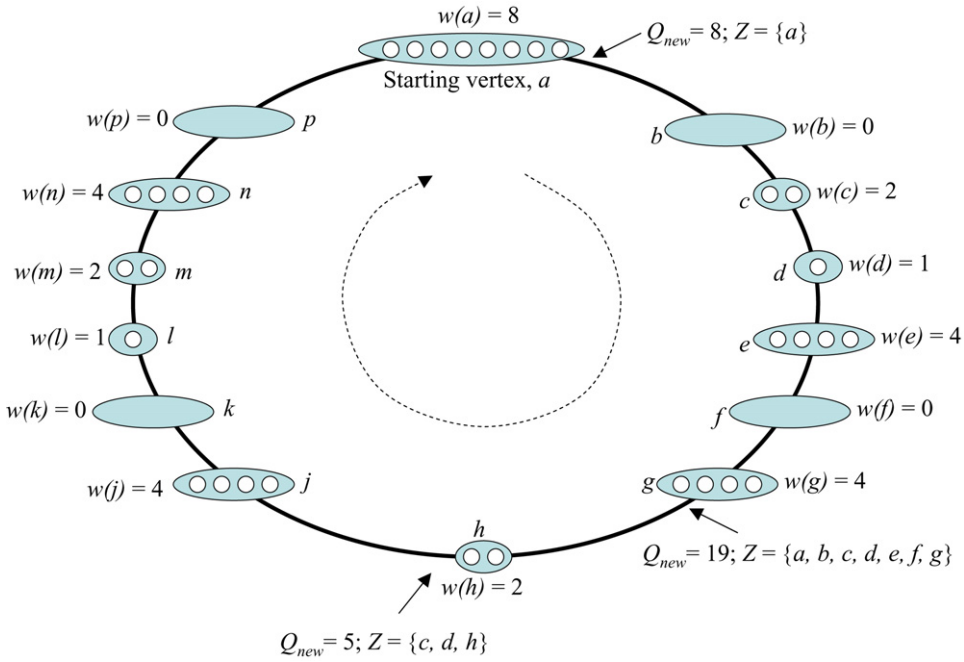

Fig. 1. Redistribution procedure for the SSBB algorithm (Lemma 3.3).

The following lemma is easier than it appears, and differs from Lemma 2.1 in the following two aspects: (i) weights of vertices in T are powers of 2, and (ii) demand from a vertex is not split.

Lemma 3.3. *Let T be a tree with each edge having capacity U, a power of two. For all v in tree T, let $w(v)$ be a power of 2 with $w(v) < U$. Then there is an efficiently computable flow on T that redistributes the weights, respecting the cable capacity and without splitting a vertex weight, so that each vertex receives a new weight $w'(j)$ that is either 0 or U. Moreover,*

$$\mathbf{Pr}\big[w'(j) = U\big] = w(j)/U.$$

Proof. Using the argument in Lemma 3.2, we find a starting vertex from which we start visiting the vertices in the directed tour (obtained by replacing each edge in T with two oppositely directed arcs) in clockwise direction with $Y = U$. The value of Q is set to 0 initially. Increment Q by $w(j)$ on visiting vertex j. Let Q_{old} and Q_{new} be the value of Q just before and after visiting a vertex, respectively. Also, maintain set Z which is initially empty. On visiting vertex j, add j to Z, and if for some integer x, $xU \in (Q_{old}, Q_{new}]$, then we do the following: (i) we find $W \subseteq Z$ such that $\sum_{i\in W} w(i) = Q_{new} - xU$, and (ii) ask the vertices in $Z \backslash W$ to send their weights to j while removing them from Z (refer Fig. 1).

We now show how to find $W \subseteq Z$. Let g be the first vertex at which $Q_{new} \geqslant U$. The proof of Lemma 3.2 would have marked g and asked g to send $Q_{new} - U$ to the next marked vertex lying clockwise on the tour. We show that there exists a $W \subseteq Z$ such that $Q_{new} - \sum_{i\in W} w(i) = U$. This is the same as showing that there exists a set $M \subseteq Z$ such that $\sum_{i\in M} w(i) = U$. Recall that no vertex in Z has a weight more than U. To show that there exists an M, all we need to do is the following. Merge two vertices $a, b \in Z$ of same weight w into one vertex with weight $2w$. Since w is a power of 2, the weight of the new vertex remains a power of 2. Continue this merging process until (i) a vertex in Z is of weight U or (ii) no more merging is possible. While the former proves our claim, the latter is not possible as it is a contradiction to $\sum_{i\in Z} w(i) \geqslant U$, because $\sum_{k=0}^{i} 2^k < 2^{i+1}$. Once M is found, $W = Z \backslash M$. The vertices in W will be the sole contributors of the flow from g to the next vertex lying clockwise on the tour. This argument holds true for every vertex j at which $xU \in (Q_{old}, Q_{new}]$ for some integer x. Notice that the probability that a vertex $j \in T$ receives (gets assigned) a weight of U is $w(j)/T$, which is exactly what we needed, as per the lemma statement.

The proof will be complete once we show that the redistribution can be done on T rather than on the directed arcs of the Euler tour on T. Consider a leaf node $l \in T$ that is in the Euler tour. Let $h \in T$ be the node that was visited just before and after l (h is l's parent in T, which is rooted at r). We use x' and x'' to represent vertex $x \in T$ in the directed Euler tour, with the tour entering x' and leaving x''. Let $f_{h'l'}$ and $f_{l''h''}$ be the flows on arcs from h' to l' and l'' to h'', respectively (refer Fig. 2). During the redistribution process, if l had sent all its weight to some vertex—lying clockwise on the tour—that was assigned a weight of U, then ask h' to send the flow $f_{h'l'}$ directly to h'' instead of sending it through l. If l was assigned a weight of U in our redistribution process, then ask the vertices in W to reroute their flow bypassing l, i.e., make the flow, from vertices in W, coming into h' go directly to h'' instead of routing it through l. Remove l from T, and repeat this process

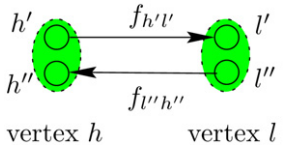

Fig. 2. l is a leaf node with h being its parent in T.

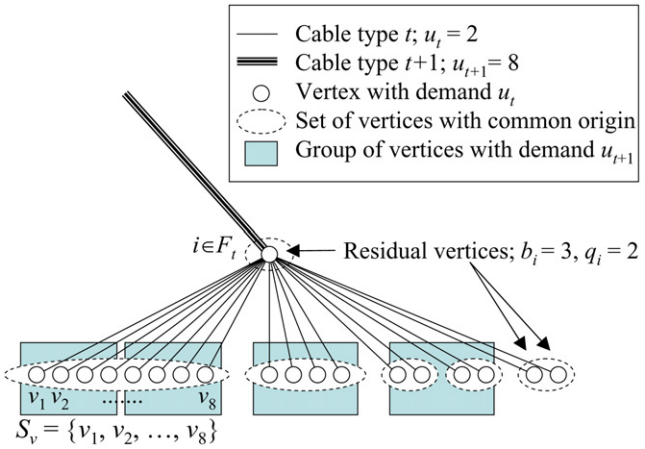

Fig. 3. Steps performed at stage t of the SSBB algorithm.

for all leaf nodes in T. Note that whenever a leaf node is removed from T, the flow on the tree edge connecting that node to T is at most U. This process stops when there is just one node left in T. This completes the proof of Lemma 3.3. □

Now that we have a new redistribution procedure in place, which requires that the demands are powers of 2, we first round the demands (of the vertices in D) upto the nearest powers of 2, and solve the problem for these new (rounded) weights. Although this means that we might install at most twice the required cable capacities, thereby losing a factor of 2 in the approximation ratio, we will have enough cable capacities installed so as to route the original demands without having to split them.

Construct a ρ-approximate Steiner tree spanning D, using cables of capacity u_1, and redistribute the demands using the construction in the proof of Lemma 3.3 with $U = u_1$ to collect *integral* demands at some subset of vertices in D. Later, vertices of integral demands are duplicated so that the demands at vertices are unit weight. Now, replace vertex $v \in D$ of weight $w(v)$ by $w(v)$ unit weight vertices. Let $S_v = \{v_1, \ldots, v_{w(v)}\}$ be the set of unit weight vertices that represent v. We call v to be the *origin* of v_i, $i = 1$ to $w(v)$. Our algorithm will ensure that the unit weight demands having a common origin travel together—along a single path—towards the sink.

The algorithm given below proceeds along the same lines as that in [7]. At the beginning of stage 1, $D_1 = D$ with each demand $j \in D$ having weight $d_j = 1 = u_1$. In general, at the beginning of stage t, D_t is a set of $|D|/u_t$ vertices, each with demand u_t. During stage t, our algorithm (presented below) uses the value u_{t+1} as the "aggregation threshold" to combine several demands of weight u_t into a single demand of weight u_{t+1}. The cables required to perform such an aggregation are bought by the algorithm. The demand will reach the root at the end of the algorithm. The final solution is then given by the union of all the paths used in the aggregation stages. Given below are the steps performed at state t of the algorithm.

S1. *Mark* each demand v in D_t with probability $p_t = u_t/g_t$, and let D_t^* be the marked demands.
S2. Construct a ρ-approximate Steiner tree T_t on $F_t = D_t^* \cup \{r\}$. Install a cable of type $t+1$ on each edge of this tree.
S3. For each vertex $j \in D_t$, send its weight $w(j)$ to the nearest member of F_t using cables of type t (refer Fig. 3). If two vertices have a common origin, ensure that both vertices send their weight to the same $i \in F_t$, as this guarantees that vertices having a common origin travel together, thus satisfying the indivisibility constraint. Let $w_t(i)$ be the weight collected at $i \in F_t$.
S4. For each $i \in F_t$, order the vertices that sent their weight of u_t to i such that the vertices in S_v are ordered before the vertices in S_u, if $|S_v| \geqslant |S_u|$.
Divide the vertices in the ordered set into groups of u_{t+1}/u_t vertices, starting from the first vertex, leaving behind $b_i = (\frac{w_t(i)}{u_t} \bmod \frac{u_{t+1}}{u_t})$ *residual* vertices at the end. Send back the weight of u_{t+1} emanating from each group of u_{t+1}/u_t vertices back from i to a random member of that group, buying new cables of type $t+1$. Since u_t, u_{t+1} and $|S_k|$, for all k, are powers of 2 by definition, our construction ensures the following: (i) set S_k, with $|S_k| \geqslant u_{t+1}/u_t$, is divided into

$p = |S_k|/u_{t+1}$ groups, and (ii) set S_k, with $|S_k| < u_{t+1}/u_t$ belongs to exactly one group. That is, vertices with common origin travel together.

S5. For each $i \in F_t$, divide the b_i residual vertices into q_i sets $R_i^1, \ldots, R_i^{q_i}$, with each set containing vertices having common origin, and the weight $w(R_i^j)$ of a set R_i^j being the number of vertices it contains. Let $F'_t = \emptyset$ initially. For each $i \in F_t$, if $b_i \geqslant 1$, then add q_i copies of i into F'_t, one for each set (origin), with each copy carrying the weight of the sum of the vertex weights in the set that it represents. Observe that the weights of the vertices in F'_t are powers of 2. Since the vertices in F'_t are mere copies of that in F_t, T_t spans all the vertices in F'_t. Use the procedure of Lemma 3.3 on T_t spanning F'_t with $U = u_{t+1}$ to aggregate residual weights into groups of weight exactly u_{t+1} in a subset of vertices in F'_t. During the redistribution procedure, for every $i \in F_t$, ensure that its copies in F'_t are visited consecutively. This, along with the fact that $u_t \sum_{j=1}^{q_i} w(R_i^j) < U = u_{t+1}$ for every $i \in F_t$ ensures that at most one copy of i in F'_t, representing $i \in F_t$, gets assigned a weight of u_{t+1}. Transform this redistribution among the vertices in F'_t into a redistribution among the vertices in F_t by assigning a weight of u_{t+1} to vertex $i \in F_t$ if one of i's copy was assigned a weight of u_{t+1} in F'_t, and 0 otherwise. Notice that the probability that a vertex $i \in F_t$ is assigned a weight of u_{t+1} still depends on i's weight (residual weight, which is $b_i u_t$). For every $i \in F_t$, that receives a weight of u_{t+1}, choose a vertex $v \in b_i$ uniformly at random, and send the weight of u_{t+1} from i to v using cables of type $t+1$.

When $t = K$, we set the probability for non-root vertices $p_K = 0$, which implies that no vertex in stage $t = K$ is marked. The weights from all the vertices in D_K are directly routed to r using cables of capacity K. The approximation analysis for our SSBB algorithm is exactly the same as that for the Gupta et al.'s DSSBB algorithm [7]. For the sake of completeness, we present the lemmas from [7] which we used to arrive at the approximation ratio.

Lemma 3.4. *(See [7].) For every non-root vertex $j \in D$ and stage t*

$$\mathbf{Pr}[j \in D_t] = 1/u_t.$$

Lemma 3.5. *(See [7].) Let T_t^* be the optimal Steiner tree on F_t, and $c(T_t^*) = \sum_{e \in T_t^*} c_e$. Then*

$$\mathbf{E}\big[c(T_t^*)\big] \leqslant \sum_{s>t} \frac{1}{\sigma_s} C^*(s) + \sum_{s \leqslant t} \frac{1}{\delta_s \cdot g_t} C^*(s). \tag{2}$$

The proof of the following lemma is given as Lemma 4.4 in [7] with $\epsilon = 1$.

Lemma 3.6. *(See [7].) The expected cost incurred in stage t is at most $(2 + \rho + \frac{2}{1+\epsilon})$ times $\sigma_{t+1}\mathbf{E}[c(T_t^*)]$, where T_t^* is the optimal Steiner tree on F_t.*

Theorem 3.1. *Our algorithm for the SSBB problem guarantees an approximation ratio of* 153.6.

Proof. We lose a factor of 2 from rounding up the weights of vertices to the nearest powers of 2. By rounding the costs and capacities of cables to powers of $1 + \epsilon = 2$ (for $\epsilon = 1$), we lose additional factor of 4 in the approximation ratio. We incur a cost of $\rho \sum_j C^*(j)/\sigma_j$ for the construction of Steiner tree T_0. The total cost C_s incurred during the algorithm proper is obtained by substituting Lemma 3.6, and summing over all t for $\epsilon = 1$. This shows that the coefficient of $C^*(s)$ is at most

$$2 \times (1+\epsilon)^2 \left(2 + \rho + \frac{2}{1+\epsilon}\right) \times \left(\sum_{t=0}^{s-1} \frac{\sigma_{t+1}}{\sigma_s} + \sum_{t \geqslant s} \frac{\delta_t}{\delta_s}\right). \tag{3}$$

Since σ_t and δ_t are powers of $1 + \epsilon$, the summations will be upper bounded by $1 + 1/\epsilon$. In the final stage, routing the demands directly to the sink has an expected cost of C^* as the demands are routed using the largest available cables (which are completely "full"), and the cost of such routing per unit of demand and unit of distance is the minimum possible by economies of scale. Thus, the cost of the final solution is at most

$$2 \times (1+\epsilon)^2 \left[\left(2 + \rho + \frac{2}{1+\epsilon}\right) \times 2\left(1 + \frac{1}{\epsilon}\right) + 1\right] C^*, \tag{4}$$

which gives a ratio of 153.6 for $\epsilon = 1$. Here, we are using the current best Steiner tree approximation algorithm, which guarantees an approximation ratio of $\rho = 1 + \ln(3)/2$ [10]. □

3.2. The DSSBB problem

Gupta et al.'s [7] algorithm guarantees a 76.8 approximation ratio for the DSSBB problem. In their algorithm, they round the costs and capacities of cables to powers of 2, i.e., ϵ was set to 1. Although Gupta et al. were of the impression that their

ratio applies to the SSBB problem, unfortunately, their approach does not guarantee that the flow from a node follow a single path to the sink. Since the flow from a node could be split along the way to its sink anyway, instead of rounding up the costs and capacities to the powers of 2, one could rather generalize the rounding to the powers to $1+\epsilon$, where $\epsilon \leqslant 1$. This minor modification to their algorithm improves the approximation ratio for the DSSBB problem.

Theorem 3.2. *The approximation ratio α_K of our DSSBB algorithm is at most* 67.94.

Proof. The proof proceeds along the same lines as that in Theorem 3.1, except that since weights of vertices are not required to be powers of 2, we save a factor of 2. The cost of the final solution is at most

$$(1+\epsilon)^2\left[\left(2+\rho+\frac{2}{1+\epsilon}\right)\times 2\left(1+\frac{1}{\epsilon}\right)+1\right]C^*, \tag{5}$$

which when optimized for ϵ gives a ratio of 67.9369 for $\epsilon \approx 0.553$. Here, we are using the current best Steiner tree approximation algorithm, which guarantees an approximation ratio of $\rho = 1+\ln(3)/2$ [10]. □

Corollary 3.1. *For a fixed K, $\alpha_2 < 17.7$, $\alpha_3 < 23.2$, $\alpha_4 < 28.8$, $\alpha_5 < 34.3$, and so on.*

Proof. By expanding the summations in Eq. (3) rather than bounding them by $1+1/\epsilon$, it can be easily shown that for a fixed K, there exists an $\epsilon > 0$ for which the corollary can be mathematically verified. □

4. Conclusion

In this paper, we presented a 153.6-approximation algorithm for the SSBB problem improving the previous best ratio of 216. For the case in which the flow is splittable, we improve the previous best ratio of 76.8 to α_K, where α_K is less than 67.94 for all K.

We believe that finding a reasonable lower bound for the SSBB problem may help in obtaining better approximation ratios. Our approach of analyzing the ratio, adapted from [7], may not be the best way to analyze the final ratio after all. It should be interesting to see whether our ratio of 153.6 for the SSBB problem can be reduced by a factor of 2 through a more careful analysis. An interesting variant of the SSBB problem would be one with an additional restriction that the final network be a tree, which would be a natural generalization of the *Capacitated Minimum Steiner Tree* (CMStT) problem [9]. To our knowledge, there is no known approximation algorithm for this variant.

References

[1] B. Awerbuch, Y. Azar, Buy-at-bulk network design, in: Proc. 38th IEEE Symposium on Foundations of Computer Science (FOCS), 1997, pp. 542–547.
[2] Y. Bartal, Competitive analysis of distributed on-line problems-distributed paging, Ph.D. Thesis, Tel-Aviv University, Israel, 1994.
[3] J. Fakcharoenphol, S. Rao, K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, in: Proc. 33rd ACM Symposium on Theory of Computing (STOC), 2003, pp. 448–455.
[4] F. Grandoni, G.F. Italiano, Improved approximation for single-sink buy-at-bulk, in: Proc. 17th International Symposium on Algorithms and Computation (ISAAC), 2006, pp. 111–120.
[5] N. Garg, R. Khandekar, G. Konjevod, R. Ravi, F.S. Salman, A. Sinha, On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem, in: Proc. 8th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2001, pp. 170–184.
[6] S. Guha, A. Meyerson, K. Munagala, A constant factor approximation for the single sink edge installation problems, in: Proc. 33rd ACM Symposium on Theory of Computing (STOC), 2001, pp. 383–399.
[7] A. Gupta, A. Kumar, T. Roughgarden, Simpler and better approximation algorithms for network design, in: Proc. 35th ACM Symposium on Theory of Computing (STOC), 2003, pp. 365–372.
[8] R. Jothi, B. Raghavachari, Improved approximation algorithms for the single-sink buy-at-bulk network design problems, in: Proc. 9th Scandinavian Workshop on Algorithm Theory (SWAT), 2004, pp. 336–348.
[9] R. Jothi, B. Raghavachari, Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design, ACM Transactions on Algorithms 1 (2) (2005) 265–282.
[10] G. Robins, A. Zelikovsky, Improved Steiner tree approximation in graphs, in: Proc. 11th ACM–SIAM Symposium on Discrete Algorithms (SODA), 2000, pp. 770–779.
[11] F.S. Salman, J. Cheriyan, R. Ravi, S. Subramanian, Approximating the single-sink link-installation problem in network design, SIAM Journal on Optimization 11 (3) (2000) 595–610.
[12] K. Talwar, Single-sink buy-at-bulk LP has constant integrality gap, in: Proc. 9th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2002, pp. 475–486.